



**android**

# LẬP TRÌNH ANDROID 2

---

## SQLITE

## ❑ Giới thiệu về SQLite

- ❖ Giới thiệu về SQLite
- ❖ Các kiểu dữ liệu trong SQLite

## ❑ Sử dụng SQLite

- ❖ SQLiteOpenHelper: Tạo database và table
- ❖ SQLiteDatabase: Xử lý CRUD
- ❖ Tổ chức cấu trúc ứng dụng



**android**

# LẬP TRÌNH ANDROID 2

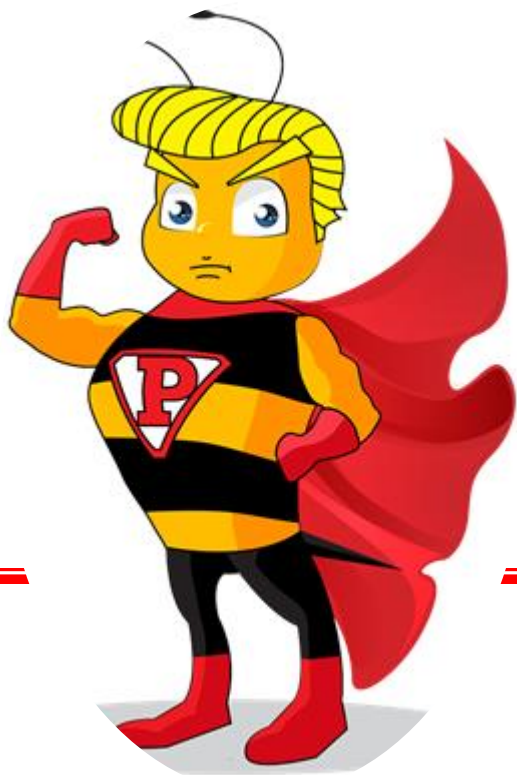
---

## SQLITE

# MỤC TIÊU

- ◎ GIỚI THIỆU VỀ SQLITE
- ◎ CÁC KIỂU DỮ LIỆU TRONG SQLITE
- ◎ SỬ DỤNG SQLITEOPENHELPER: TẠO DATABASE VÀ TABLE
- ◎ SỬ DỤNG SQLITEDATABASE: XỬ LÝ CRUD
- ◎ TỔ CHỨC CẤU TRÚC ỨNG DỤNG





# GIỚI THIỆU VỀ SQLITE

---

...

- ❑ SQLite là phần mềm quản lý cơ sở dữ liệu SQL nhưng SQLite không có máy chủ riêng biệt để xử lý.
- ❑ Đặc điểm:
  - ❑ Đơn giản & gọn nhẹ: Dữ liệu database được lưu vào một file duy nhất, không cần cài đặt, không cần cấu hình.
  - ❑ Không có user, password hay quyền hạn trong Sqlite



- ❑ SQLite không cần thiết phải cài đặt, cấu hình, không cần mô hình client- server để hoạt động.
- ❑ Một SQLite Database đầy đủ được lưu giữ trong một disk file đơn. SQLite là rất nhỏ gọn, nhỏ hơn 400kB đã được cấu hình đầy đủ hoặc nhỏ hơn 250kB khi đã bỏ qua các tính năng tùy ý.
- ❑ Các Transaction trong SQLite là tuân theo đầy đủ chuẩn ACID, đảm bảo truy cập an toàn từ nhiều tiến trình hoặc thread.
- ❑ SQLite hỗ trợ hầu hết các tính năng của một ngôn ngữ truy vấn trong chuẩn SQL92.



## ❑ Lớp lưu trữ (Storage Class)

Mỗi giá trị được lưu giữ trong CSDL SQLite có một trong các lớp lưu trữ (Storage Class) sau:

Lớp lưu trữ	Miêu tả
NULL	Giá trị là một giá trị NULL
INTEGER	Giá trị là một số nguyên có dấu, được lưu giữ trong 1, 2, 3, 4, 6, hoặc 8 byte tùy thuộc vào độ lớn của giá trị
REAL	Giá trị số thực dấu chấm động, được lưu giữ như là một số thực dấu chấm động 8-byte IEEE
TEXT	Giá trị là một text string, được lưu trữ bởi sử dụng Encoding của cơ sở dữ liệu (UTF-8, UTF-16BE hoặc UTF-16LE)
BLOB	Giá trị là một blob của dữ liệu, nhập vào như thế nào thì lưu giữ chính xác như thế



## □ Kiểu lưu trữ (Affinity Type)

**Affinity Type** trên các cột. Bất cứ cột nào có thể vẫn lưu giữ bất kỳ kiểu dữ liệu nào nhưng lớp lưu trữ ưu tiên cho một cột được gọi là **Affinity** của nó.

Affinity	Miêu tả
TEXT	Cột này lưu giữ tất cả dữ liệu sử dụng các lớp lưu trữ NULL, TEXT hoặc BLOB
NUMERIC	Cột này có thể chứa các giá trị sử dụng tất cả 5 lớp lưu trữ
INTEGER	Vận hành giống như một cột với NUMERIC affinity với một ngoại lệ trong một biểu thức CAST
REAL	Vận hành giống như một cột với NUMERIC affinity, ngoại trừ rằng nó ép các giá trị nguyên thành dạng biểu diễn số thực dấu chấm động
NONE	Một cột với NONE affinity không ưu tiên một lớp lưu trữ nào khi so với lớp khác và không ép dữ liệu từ một lớp lưu trữ này sang dạng một lớp lưu trữ khác

## □ Kiểu dữ liệu

Kiểu dữ liệu	Affinity
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 , INT8	INTEGER
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT	TEXT

Kiểu dữ liệu	Affinity
BLOB •Không có kiểu dữ liệu nào được xác định	NONE
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC

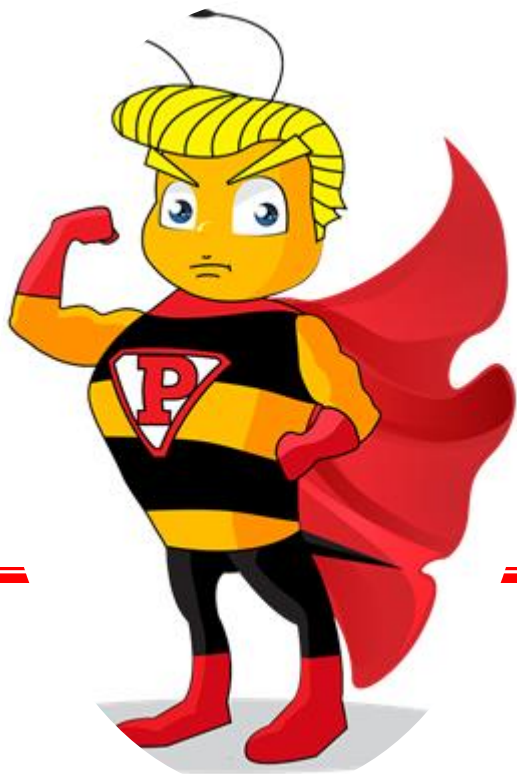
## ❑ Kiểu dữ liệu Boolean trong SQLite

- ❖ SQLite không hỗ trợ lớp lưu trữ Boolean.
- ❖ Giá trị Boolean (false/ true) được lưu trữ dưới dạng các số nguyên: 0/1.

## ❑ Kiểu dữ liệu Date và Time trong SQLite

- ❖ SQLite không hỗ trợ lớp lưu trữ date/time.
- ❖ Giá trị Date/Time được lưu trữ dưới dạng giá trị TEXT, REAL hoặc INTEGER.

Method	Description
<code>getColumnNames()</code>	This method is used to get the Array of column names of our SQLite table.
<code>getCount()</code>	This method will return the number of rows in the cursor.
<code>isClosed()</code>	This method returns a Boolean value when our cursor is closed.
<code>getColumnCount()</code>	This method returns the total number of columns present in our table.
<code>getColumnName(int columnIndex)</code>	This method will return the name of the column when we passed the index of our column in it.
<code>getColumnIndex(String columnName)</code>	This method will return the index of our column from the name of the column.
<code>getPosition()</code>	This method will return the current position of our cursor in our table.



# SỬ DỤNG SQLITE

---

...

Android cung cấp hai Class hỗ trợ cho việc tạo và quản lý Database:

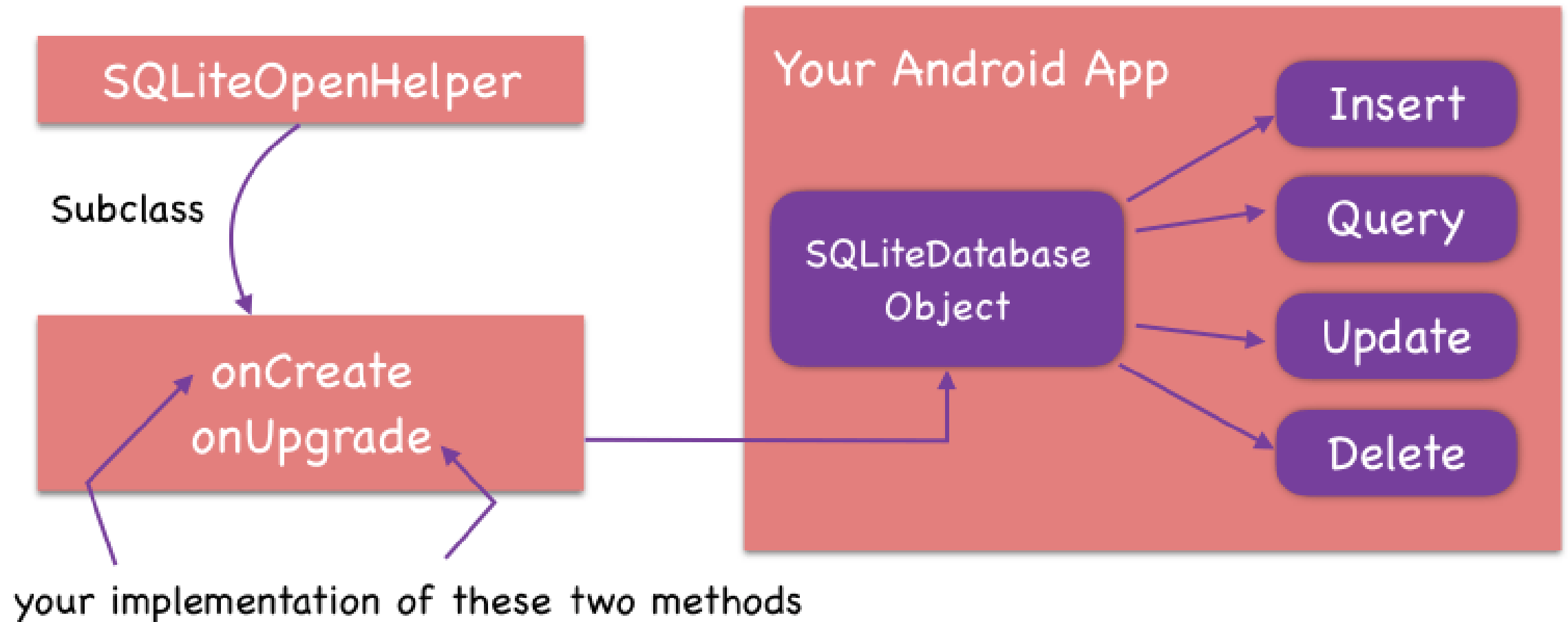
## ☐ **SQLiteOpenHelper**

- ☐ Là lớp abstract chịu trách nhiệm tạo/nâng cấp CSDL SQLite và trả về phiên bản đối tượng SQLiteDatabase.
- ☐ Phải tạo lớp kế thừa của lớp **abstract** này

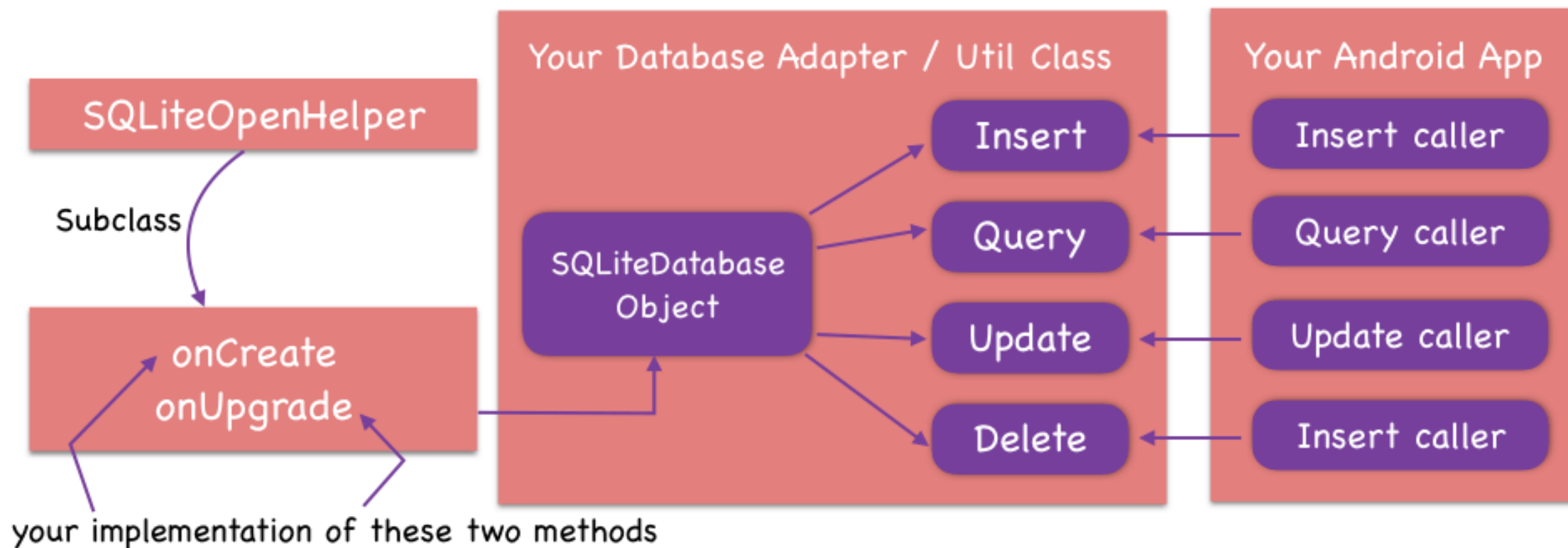
## ☐ **SQLiteDatabase**

- ☐ Là một lớp có API để xử lý các thao tác CSDL như: Create, Read, Update, Delete (CRUD)

## ❖ Cách 1: Gọi trực tiếp API từ ứng dụng



## ❖ Cách 2: Tạo một Adapter/ Util (DAO) để đóng gọi các phương thức







**FPT POLYTECHNIC**



**android**

# LẬP TRÌNH ANDROID 2

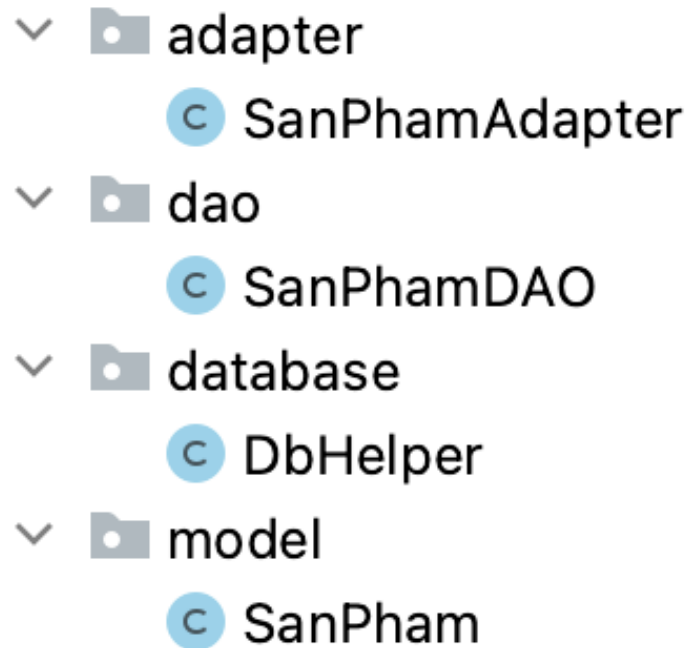
---

SỬ DỤNG SQLITE

[www.poly.edu.vn](http://www.poly.edu.vn)

❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm

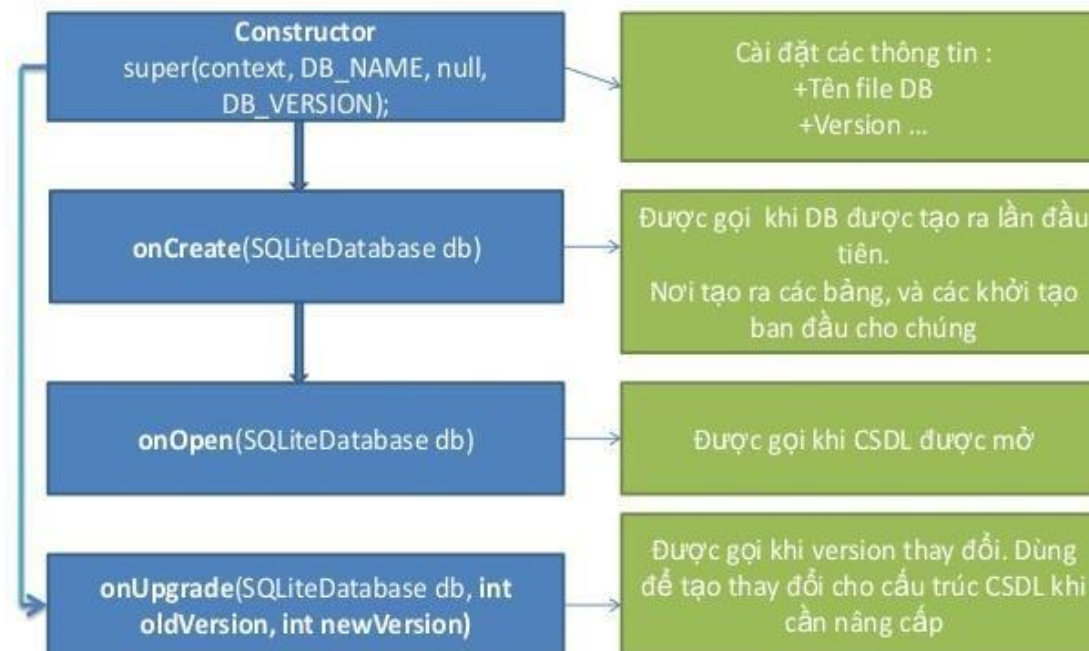
➤ Để dễ viết code và bảo trì chúng ta cần tổ chức cấu trúc ứng dụng theo nhiều **package**



```
graph TD; adapter[adapter] --> SanPhamAdapter[SanPhamAdapter]; dao[dao] --> SanPhamDAO[SanPhamDAO]; database[database] --> DbHelper[DbHelper]; model[model] --> SanPham[SanPham];
```

- ▼ adapter
  - SanPhamAdapter
- ▼ dao
  - SanPhamDAO
- ▼ database
  - DbHelper
- ▼ model
  - SanPham

- ❖ Tạo Class kế thừa SQLiteOpenHelper và override 2 phương thức onCreate và onUpgrade
  - **onCreate**: Được gọi khi CSDL chưa tồn tại. Nếu CSDL không tồn tại được gọi tự động.
  - **onUpgrade**: Thực hiện khi có phiên bản CSDL mới. Khi ta nâng version mới, cần xóa các table cũ và gọi lại onCreate



## ❖ Ví dụ: Tạo database và table quản lý thông tin sản phẩm

- Tạo Hàm khởi tạo **DbHelper**
- Tạo database "**QLSP**" với version **1**

```
public class DbHelper extends SQLiteOpenHelper {  
  
    1 usage  
    public DbHelper(Context context){  
        super(context, name: "QLSP", factory: null, version: 1);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {...}  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {...}  
}
```

❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm

➤ Hàm **onCreate**: Thực hiện viết code tạo table và thêm data vào table

```
@Override
public void onCreate(SQLiteDatabase db) {
    //tạo bảng sản phẩm
    String qSP = "CREATE TABLE SANPHAM (masp integer primary key autoincrement, tensp text, gia integer)";
    db.execSQL(qSP);

    //thêm data cho bảng sản phẩm
    String data = "INSERT INTO SANPHAM VALUES(1, 'bánh', 10000), (2, 'kẹo', 5000)";
    db.execSQL(data);
}
```

❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm

➤ Hàm **onUpgrade**: Thực hiện xóa và gọi lại onCreate nếu có version mới.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if(oldVersion != newVersion){
        db.execSQL("DROP TABLE IF EXISTS SANPHAM");
        onCreate(db);
    }
}
```

- ❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm
  - Viết các hàm thực hiện CRUD (sử dụng SQLiteDatabase)
    - Tạo constructor cho class DAO

```
public class SanPhamDAO {  
    5 usages  
    DBHelper dbHelper;  
    1 usage  
    public SanPhamDAO(Context context){  
        dbHelper = new DBHelper(context);  
    }  
}
```

## ❖ Ví dụ: Tạo database và table quản lý thông tin sản phẩm

### ➤ Viết các hàm thực hiện CRUD (sử dụng SQLiteDatabase)

#### ■ Lấy danh sách sản phẩm

```
//get danh sách
1 usage
public ArrayList<SanPham> getDS(){
    ArrayList<SanPham> list = new ArrayList<>();
    SQLiteDatabase sqLiteDatabase = dbHelper.getReadableDatabase();
    Cursor cursor = sqLiteDatabase.rawQuery( sql: "SELECT * FROM SANPHAM", selectionArgs: null);
    if(cursor.getCount() > 0){
        cursor.moveToFirst();
        do{
            list.add(new SanPham(cursor.getInt( columnIndex: 0), cursor.getString( columnIndex: 1), cursor.getInt( columnIndex: 2)));
        }while (cursor.moveToNext());
    }

    return list;
}
```



- ❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm
  - Viết các hàm thực hiện CRUD (sử dụng SQLiteDatabase)
    - Thêm một sản phẩm mới

```
//thêm 1 sản phẩm mới
public boolean themSP(SanPham sanPham){
    SQLiteDatabase sqLiteDatabase = dbHelper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("tensp", sanPham.getTensp());
    contentValues.put("giasp", sanPham.getGia());
    long check = sqLiteDatabase.insert(table: "SANPHAM", nullColumnHack: null, contentValues);
    return check != -1;
}
```

- ❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm
  - Viết các hàm thực hiện CRUD (sử dụng SQLiteDatabase)
    - Thay đổi thông tin một sản phẩm

```
//chỉnh sửa thông tin 1 sản phẩm  
public boolean thayDoiSP(SanPham sanPham){  
    SQLiteDatabase sqLiteDatabase = dbHelper.getWritableDatabase();  
    ContentValues contentValues = new ContentValues();  
    contentValues.put("tensp", sanPham.getTensp());  
    contentValues.put("giasp", sanPham.getGia());  
    long check = sqLiteDatabase.update(table: "SANPHAM",  
                                       contentValues,  
                                       whereClause: "masp=?",  
                                       new String[]{String.valueOf(sanPham.getMasp())});  
    return check != -1;  
}
```

- ❖ **Ví dụ:** Tạo database và table quản lý thông tin sản phẩm
  - Viết các hàm thực hiện CRUD (sử dụng SQLiteDatabase)
    - Xóa một sản phẩm

```
//Xóa 1 sản phẩm
public boolean xoaSP(int masp){
    SQLiteDatabase sqLiteDatabase = dbHelper.getWritableDatabase();
    long check = sqLiteDatabase.delete(table: "SANPHAM",
                                       whereClause: "masp=?",
                                       new String[]{String.valueOf(masp)});
    return check != -1;
}
```



**FPT** Education

FPT POLYTECHNIC

**Thank you**